

WEB BROWSER EXTENSION AND METHOD FOR PROCESSING  
DATA CONTENT OF WEB PAGES

Field of Invention

The present invention relates to extending Web Browser capabilities and in particular to assisting a user of a Web Browser with analysis and interpretation of the data content of displayed Web pages.

Background

The majority of the content of Web pages consists of text, images, and other media together with the HyperText Markup Language (HTML) or other markup language codes which describe how the media content should be displayed by a Web Browser. This content is typically static and cannot be manipulated by the user.

Some Web pages also contain embedded active components that support manipulation of content on a specific page within constraints defined by the page designer. Examples of such components which are integrated within individual Web pages and limited to those pages include Java applets and ActiveX controls. Many controls of this kind impose a per-page download penalty that can impact the performance of the Web page. Another type of embedded active component is plug-in applications for Netscape's Navigator Web Browser, which are each associated with a specific type of non-HTML data content and allow users to 'view' or experience that content. Example plug-ins are Adobe's Acrobat document viewer and Macromedia's Shockwave interactive animation and sound player.

Other functions for working with Web pages have been implemented recently as Browser extensions. These have been used, for example, to reserve user interface space within the Browser for displaying links to other Web pages or links to other applications. The Alexa Browser extension performs a database lookup using the URL of the current Web page and displays a variety of content related to this URL. Typically, linked applications are stand-alone and have no particular relationship to the Browser content once launched and include no mechanism for communicating back to the Browser. In other cases, the browser extensions have enabled simple manipulation of Web page content from context menus, such as applying a background colour to a selected block of text to highlight it, or listing and displaying the images or the links in the current Web page. In all of these cases, no analysis or interpretation of the semantics of the Web page data content is performed. Furthermore, to

the extent that a user is able to select a portion of data content to be manipulated, that portion of content is typically manipulated as a block without separation of individual data items within the selected portion of content.

AltaVista Babel Fish is a Web page content translation tool. The user links to the Babel Fish Web site by clicking on the Babel Fish icon in the toolbar of Microsoft's Internet Explorer 5. The user chooses a language and specifies a URL or selected text via an intermediate screen (typing in data or using standard copy and paste functions) and then the translation is performed on AltaVista's relevant server computer. While the Babel Fish tool adds to the capabilities of a Web Browser and provides a method for manipulation of Web page content, the only new component implemented on the Web client computer is a mechanism for navigating to the Babel Fish site. Note that Babel Fish does not perform the translation processing on the data processing apparatus which is running the Web Browser (i.e. on the Web client) but on a remote server. After the tool has been used, the user has to navigate back to the original Web page. Functions as computationally complex as translation require more processing power than is typically available on a Web client device and considerable storage space is required to hold dictionaries in multiple languages, and so there are significant technical reasons against implementing Babel Fish translation on a Web client device.

Thus, existing Web Browsers, embedded components, Browser extensions and other associated applications have significantly limited capabilities for manipulation of Web page data content and in particular for analysis and interpretation of semantic data content and for manipulation of individual data items of a set of items selected from a Web page by a user.

#### Summary of Invention

In a first aspect, the invention provides a set of components for cooperating with a Web Browser to process the semantic data content of Web pages, wherein the Web Browser implements functions to display Web page contents on a display apparatus, and wherein the set of components are operable to run on a Web client data processing apparatus together with the Web Browser, the set of components including: a data content transfer component, responsive to user selection of a desired semantic processing operation and user selection of a set of data items within a displayed Web page, for extracting the set of data items from the Web page; and a set of data content processing components, each for processing individual data items of a set of data items extracted from any one of multiple Web pages, for performing at the Web client data processing apparatus the selected processing operation on the individual

items of the selected set, to generate a result which is dependent on the semantics of the selected set of data items; wherein the data content transfer component includes means for providing the generated result to the Web Browser for presentation to the user.

The invention extends a Web Browser by providing the capability to process semantic data content of Web pages - this means performing processing operations on actual data values where the values of the data (e.g. number values or the meaning of text words) are relevant to the result of the operation. The invention helps users to analyze and interpret text or numeric content displayed by the Browser. This goes beyond the capabilities of existing Browsers which typically only include functions for accessing Web pages and parsing and rendering Web page content for display to a user. It also goes beyond the capabilities of known Web Browser extensions which typically only provide links to related Web pages or other Web pages of interest to the user, or which provide links to other applications without providing any ability to process the semantic data content of a Web page or to return the results of the processing to the Browser.

The set of processing components are each generically applicable to multiple Web pages (preferably working with any HTML-based content on any Web page, or any markup language) and so are distinguished from Java applets, ActiveX controls and other known active components which are embedded within a single, specific Web page and perform operations on the data within a specific area of that Web page only within constraints defined by the Web page designer. This generic applicability is achieved by implementing the data content transfer component at the Browser level rather than at the page level.

The set of processing components are 'client-side tools', running on the data processing apparatus on which the Web Browser runs. This distinguishes the components of the present invention from a tool which requires a processing request and the client data to be sent across an Internet connection to a remote server which processes the data remotely and then returns the results of the processing to the client system. Preferred embodiments of the invention therefore avoid the delay and scope for failures inherent in network transmission and no per-page download overhead applies.

Although the processing operation is performed locally, some implementations of the present invention may include a retrieval of information from a remote computer for use in the processing. One example is retrieval of current taxation rates from a remote database for use in calculating tax payable on net prices listed in a displayed Web page.

The components of the invention preferably provide the results of their processing to the Web Browser to display as modified HTML content (or XML, WML or other content-description or 'markup' language of the original content), the result being represented in a pop up window or via some form of annotation of the Web page. Alternatively, the result may be displayed elsewhere in the Browser's window outside the area reserved for display of the Web page. Note that these are examples of the processing results being displayed so as to modify the information displayed by the Browser while still displaying the Web page which was displayed when the processing component and Web page data contents were selected.

A Web Browser suitable for implementation of the present invention may comprise a tool for responding to user interactions by sending to a Web server HTTP or other requests for Web pages and for presenting retrieved pages to the user (for example, providing functions for submitting requests to an Internet search engine located elsewhere in the network and for parsing and rendering received HTML). In the present application, the term 'Web Browser' is also intended to include any markup language viewer component which implements functions for displaying markup language content to a user of a data processing apparatus, and which may or may not include other functions implemented by conventional Web Browsers. The term 'Web client apparatus' should be interpreted accordingly, and may include communications-enabled PDAs, telephones with data processing capability, in-car devices, television sets and other devices. A Web client apparatus in this context may also be a device providing non-visual interfaces for accessing Web page content such as using a voice-enabled Browser to access VoiceXML content. Additionally, 'Web Browsers' suitable for implementing the invention may include significant additional functionality, such as data manipulation functions.

The present invention preferably provides a generic enablement of the Browser to work with any one of a plurality of data processing tools which are able to exchange data with the Browser-enabling code, and is therefore open-ended. That is, additional processing components can be added to this infrastructure after the enabling code has been installed on a client device. The individual processing components can typically be implemented as very small code components, suitable for storing and running on a client device with limited storage.

The present invention also provides significant advantages over the invention described in US patent No. 5,896,491, which is incorporated herein by reference. US 5,896,491 describes a set of generic, client-side tools for performing processing operations in response to user selection of an operation and selection of a set of an application's data items and applying emphasis such as highlighter colours to particular data values

to aid interpretation and analysis. A limitation of the invention of US 5,896,491 was that it required each application program to be engineered to support interoperability with the generic tools. By contrast, the set of processing components of a Web Browser according to the preferred embodiment of the present invention can access and manipulate data content in any typical HTML-based Web page, including applications built using HTML, (or another markup language).

According to a preferred embodiment of the present invention, numbers and strings can be extracted from unstructured Web content and the extracted data is then passed in a structured form to a processing component. Thus, processing components which are adapted to handle structured, numerical, tabular data are enabled to work with unstructured data in Web pages.

In a second aspect of the invention, there is provided a method for processing the semantic data content of Web pages displayed on a display apparatus via a Web Browser, the method including the following steps performed on a Web client data processing apparatus: in response to user selection of a desired semantic processing operation, and user selection of a set of data items within a displayed Web page, extracting the set of data items from the Web page; and performing the selected processing operation on the selected set of data items to generate a result which is dependent on the semantics of the selected set of data items, using a data content processing component which is capable of processing the data content of any one of multiple Web pages; providing the generated result to the Web Browser for presentation to the user.

In a third aspect, the invention provides a software component for cooperating with a Web Browser, the software component controlling the operation of a data processing apparatus on which it runs to perform the following steps in response to user selection of a desired data processing operation and user selection of a portion of a Web page displayed via the Browser: extracting the selected portion of the markup language Web page content from Web page; separating individual data items from markup tags within the selected portion; passing the separated data items to a data processing component to perform a processing operation on the individual data items; and in response to receipt by said software component of a result of said processing operation, generating a modified version of the portion of the markup language Web page content; and providing said modified portion to the Browser to replace the extracted portion of the displayed Web page.

This extraction of a selected portion of a Web page's markup language content, followed by separation of individual data items for processing, and then generation of a modified portion of markup language

to replace the extracted portion is distinguished from the prior art manipulation of blocks of text which does not involve an identification of the separate data items. In addition to a single word or number, an 'individual data item' in this context may comprise a set of tokens which form a single logical unit, such as a name combining a forename token and a surname token. The majority of Web page content is currently written in HTML, but the invention can be applied to any markup language.

This aspect of the invention of extraction of individual data values from a selected portion of a Web page's markup content, sending to a processing component, and generating modified markup content in response to receipt of a processing result is applicable to many different types of processing operation. Such operations may be, for example, a table lookup operation to obtain information associated with a data item, a chart generation operation, or a mathematical calculation.

The present invention may be implemented in a computer program product comprising computer readable program code recorded on a computer readable recording medium. Alternatively, the program code may be provided on a Web server computer in a form downloadable to Web client data processing systems. The invention can be integrated within a Web Browser as well as being provided as a component or set of components for interoperating with a Browser.

The invention according to a further aspect comprises a data processing system including a set of software components for cooperating with a Web Browser to enable processing of individual data items of a set of data items extracted from the data content of Web pages.

#### Brief Description of Drawings

Embodiments of the present invention will now be described in more detail, by way of example, with reference to the accompanying drawings in which:

Figure 1 is a schematic representation of the main software components, interface objects and their relationships, according to an embodiment of the invention; and

Figure 2 is a screen shot image showing an example implementation of the present invention in which the user interface includes an explorer bar and palette for user interaction, and in which the selected 'News Reader' operation has identified particular categories of company and generated emphasis results (background colours) which have then been applied to the displayed company names, and generated annotation results which have been displayed on a Browser status line;

### Detailed Description of Preferred Embodiment

A first embodiment of the invention uses a new type of Browser extension to Microsoft's Internet Explorer 5 Web Browser to provide a mechanism linking the selection model of the Browser to a plurality of data content processing components known as SmartMarkers. The invention may be implemented in any data processing apparatus which is suitable for use as a Web client. Such a device is capable of running a Web Browser and typically includes a processor, memory for storing data and program code (which may include an internal hard disk device or an external storage device), a display device connected to receive data via a display controller, one or more input devices (typically a mouse and a keyboard) and an input controller, and appropriate internal and external communications resources. The SmartMarker data content processing components are preferably implemented as Java(TM) program components. SmartMarkers are general purpose end-user tools for analysing and interpreting text or numeric content displayed by a Browser or SmartMarker-enabled application program. The design of SmartMarkers and a number of specific examples is described in UK patent application No. GB 9816098.9 (corresponding to PCT/GB99/02433), which is incorporated herein by reference and which includes a description of annotating application data values as a way of presenting the results of processing by SmartMarker tools. A detailed description of the design of SmartMarkers will be given later.

#### **Browser Extensions**

Browser extensions for Microsoft's Internet Explorer 5 implement a standard API (Application Programming Interface) to enable developers to reserve space within the Browser's window for displaying links to additional functions and to access the Browser's content through its Document Object Model (DOM). In particular an area of the window is typically reserved as a so-called "Explorer Bar". The DOM is a programming interface specification which lets a programmer create and modify HTML pages and XML documents as program objects. The DOM describes the logical structure of documents and the way a document is accessed and interacted with. Individual HTML elements within a document are separately addressable objects having a separate representation within the DOM, and each object is accessible via its pointer. Navigating to a desired object involves navigating through a chain of object pointers.

Known uses of Browser extensions have included use as a mechanism for linking to other applications running on a remote server computer. They have also been used to enable an application to interact with Web page data content, but only in very limited ways such as highlighting a block of text or listing and displaying images. There has previously been

no generic mechanism on a Web client system providing the capability to process the semantic data content of all HTML-based Web pages.

#### **SmartMarker-enabled Browser**

Referring to Figure 1, the first embodiment of the invention comprises the following software components: a Web Browser 10; a SmartMarker Browser extension Dynamic Link Library (DLL) 30; a SmartMarker user interface DLL 40; a SmartMarker transformation layer 70; and a plurality of SmartMarker components 80 and their respective SmartMarker algorithms 90. The SmartMarker Browser extension DLL and SmartMarker user interface DLL together provide access to the Web page data content 20 via the Browser's Document Object Model, enabling extraction of the data values of a set of data items selected by the user and enabling the extracted values to be sent to the SmartMarker transformation layer. The Smartmarker transformation layer interoperates with a SmartMarker component and associated SmartMarker algorithm to process the extracted data values. The results of the processing are returned to the SmartMarker user interface DLL 40 (via the transformation layer 70) which processes the results and applies them to the data content in the Browser. This will be described below in more detail.

The SmartMarker Browser extension DLL 30 implements the required ActiveX interfaces and services for an ActiveX Browser extension DLL, reserves user interface space within the Browser's window, acquires an ActiveX interface pointer to the Document Object Model for the Browser to provide access to the Browser's displayed data content, creates an instance of the SmartMarkers user interface ActiveX DLL 40, and displays the user interface defined by the user interface DLL in the reserved space.

The SmartMarker user interface DLL 40 is used in the creation of an instance of a SmartMarker Explorer Bar 50 and a SmartMarker palette 60 (see below), as well as its use in the accessing and extraction of data values from the Browser's data content and processing and applying the results of SmartMarker processing.

The SmartMarker Browser extension DLL 30 and SmartMarker user interface DLL 40 are implemented as self-registering ActiveX DLLs. Installation of the SmartMarkers Browser extension therefore involves the SmartMarker Browser extension DLL 30 and the SmartMarker user interface DLL 40 registering themselves in the Windows registry using standard methods. When the Browser is started it looks in the Windows registry for any Browser extension DLLs and lists those extensions on its Explorer Bar menu (accessible by selecting 'View' then 'Explorer Bar' from within the Browser). If the SmartMarker Browser extension is found, a SmartMarkers entry is added to the menu. This registry lookup and adding to the



Explorer Bar menu is standard implementation of the Microsoft Browser extension architecture.

A user can interact with the Browser using standard techniques to navigate to a Web page of interest. An input controller controls the position of a pointer on the display device in response to signals from the input devices, and the Browser responds to mouse related messages in order to identify and interpret selection of data items.

When a user selects SmartMarkers from the 'View' - 'Explorer Bar' menu, program code within the SmartMarker Browser extension DLL is invoked to:

- o reserve user interface space within the Browser's window for an Explorer Bar;
- o create an instance of the SmartMarkers Explorer Bar 50 using functions within the SmartMarker user interface DLL 40;
- o display the SmartMarkers Explorer Bar graphical user interface 50 in the reserved space;
- o access the DOM pointer for the Browser window;
- o pass the DOM pointer to the SmartMarker Explorer Bar 50 to provide access to the content 20 of the Browser window.

(Note that the Browser interface could alternatively be customised to enable the SmartMarker user interface to be displayed without using an explorer bar. Other options include tool bar buttons, a dedicated tool bar, menu options and pop-up windows.)

When displayed, the SmartMarker Explorer Bar 50 provides a first clickable button to enable the user to request display of a palette listing selectable SmartMarker operations. The SmartMarker Explorer Bar also provides a second button enabling the user to activate processing by a selected SmartMarker component of selected data content of a Web page displayed by the Browser. The SmartMarkers Explorer Bar will be displayed in subsequent Browser sessions until removed by the user by using the 'View' - 'Explorer Bar' menu.

When the user presses the 'Show SmartMarker palette' button in the SmartMarker Explorer Bar 50, program code within the SmartMarker user interface DLL 40 creates an instance 60 of a SmartMarker palette object and displays in a separate window a user interface provided by the SmartMarker palette object. The SmartMarker palette user interface lists available SmartMarker operations and listens for selection events, enabling the user to activate and deactivate a SmartMarker processing component, and then displays a key to help the user to interpret results

calculated by the activated SmartMarker and a button for requesting display of help for the activated SmartMarker.

When the user selects an individual SmartMarker operation from the list displayed in the SmartMarker palette 60, the respective SmartMarker object 80 is activated. If the user has not already done so, they then navigate to a Web page of interest. If the user then selects a block of data content in the Browser's window (using the standard Browser user interface for selection) and presses the 'SmartMark selected text' button in the SmartMarker Explorer Bar 50, the SmartMarker Explorer Bar 50 passes the DOM pointer for the Browser to the SmartMarker user interface DLL 40 and program code within the SmartMarker user interface DLL 40 is invoked to:

- o navigate from the DOM pointer for the Browser 10 to the DOM pointer for the selected content within the current document (i.e. stepping through a chain of pointers to addresses for particular objects from the Browser window object to the document object, and so on until the pointer to the selected data content is reached);
- o copy the HTML string corresponding to the selected data content into an appropriate area of memory (or alternatively, in a non-Java implementation, store the pointer);
- o parse the HTML string corresponding to the selected content to separate tokens such as words and numbers from HTML tags (implementation of parsing to strip values from strings is known in the art);
- o parse the separated tokens to strip trailing punctuation and other extraneous characters ("noise" such as currency symbols and thousands separators); and
- o send the tokens to the SmartMarker transformation layer.

When the SmartMarker transformation layer receives the 'cleaned' tokens, it formats and filters those tokens to meet the needs of the SmartMarker algorithm associated with the active SmartMarker. For example, some SmartMarker algorithms require only numeric data, while others require only string data. After formatting, the tokens are sent to the algorithm to be processed. This step is standard processing for SmartMarkers.

When the SmartMarker algorithm receives the tokens, it retrieves information about the current SmartMarker (obtaining its detailed properties) and then executes the method *Algorithm.calculateResults* to perform calculations on the data values of the tokens. The calculation is specific to the individual SmartMarker processing operation. For example, a SmartMarker algorithm might compare each token to a stored list of known keywords such as the names of partners or competitors, and

highlight matches. A second example is a spell checker. A third example is to compare each token to a database or dictionary of meanings of acronyms, or technical, legal, or business terms, and to annotate individual words when the mouse pointer overlays them. A fourth example is to access from a database, and to display in pop-up windows, information about products, companies or individual people identified within a Web page - the name of the product, company or person can be used as a key for a database lookup operation. A further example SmartMarker operation is to draw a business chart to aid analysis of numbers in a table or embedded in free-form text.

The results of the calculations are in two forms: emphasis results and annotation results. This performance of calculations to generate results which are relevant to the semantics of the data is standard SmartMarkers processing.

Emphasis results are used to highlight individual tokens. For example, a different emphasis might be used to highlight partners from that used to highlight competitors. Annotation results are used to provide additional information about a token. For example, in response to a mouse moving over the name of a competitor within a processed portion of text, a brief description of that competitor company might be displayed.

The results are returned to the SmartMarker user interface DLL through the SmartMarker transformation layer, and `endResults()` is called to notify the user interface DLL when a complete set of results has been provided. On receipt, these results are processed as follows:

- For each token that has any results, an HTML `<SPAN>...</SPAN>` tag pair is constructed to wrap that token. The `<SPAN>...</SPAN>` tag set is a standard HTML tag set used to separate one section of a Web page from another. It enables application of a set of attributes such as color or font to a range of text between the tag pairs. Constructing these tags for individual tokens enables the setting of specific attributes representing the results of the SmartMarker processing for individual number values or text words. In the known prior art, it is not typical for individual words or number values within a page of HTML to have an HTML `<SPAN>...</SPAN>` tag pair associated with them.
- For emphasis results, a style parameter is added to the `<SPAN>` tag in the form `STYLE='background-color:#rrggbb'` where rr, gg, bb represent the red, green, and blue components of a color calculated for this token by the SmartMarker transformation layer from the emphasis result calculated by the SmartMarker algorithm. The style

parameter is a standard CSS (Cascading style sheet) attribute recognized by the web browser as a request to change the presentation of a block of content.

- 5       ○ For annotation results, a segment of dynamic HTML is added to the <SPAN> tag in the form `onmouseover=' window.status="xxx"' onmouseout='window.status=""` where xxx represents the annotation string calculated for this token by the SmartMarker algorithm. The dynamic HTML is a standard tag attribute recognized by the Web
- 10       Browser as a request to display the specified string in the browser status area when the user moves the mouse over this token and to clear the status area when the mouse is moved elsewhere. The Browser status area is a fixed area of the Browser window reserved for status messages.
- 15       ○ As noted previously, annotation results may equally be presented as pop-up boxes ('Hover Help' style) by adding `TITLE='xxx'` to the SPAN tag pair, instead of status line messages.
- 20       When the results have been processed, the SmartMarker user interface DLL applies the results to the content in the Browser as follows:
  - 25       ○ The tokens wrapped in <SPAN>...</SPAN> tag pairs are merged with the copy of the original HTML of the selected data content which is held in memory, by replacing the original unwrapped tokens to create a replacement HTML string.
  - 30       (In an alternative non-Java implementation, if no copy of the extracted HTML was stored in memory and only a pointer was stored, a set of replacement HTML strings comprising the wrapped tokens is associated with the stored pointer).
  - 35       ○ A standard document object model service (pasteHTML) is then called to paste the replacement HTML string into the currently selected Web page.
  - 40       ○ The Browser immediately reinterprets the replaced HTML and honours the <SPAN> tag parameters by applying background color and mouse-over support to the tokens modified in response to emphasis and annotation results calculated by the algorithm.

Thus, from a user's perspective, selection of a specific SmartMarker operation from a list displayed in a palette, selection of a set of data items such as a block of text or a set of numerical values from a Web page, and pressing the 'SmartMark selected text' button simply

results in the selected area of the Web page being updated with highlights and annotations.

Figure 2 shows an example display screen including a Browser window 100, SmartMarker explorer bar 50 including selectable buttons 110, and a SmartMarkers palette 60 displayed in a separate window. The user has selected the 'Show SmartMarker palette' button within the explorer bar, then selected the 'News Reader' SmartMarker operation from the palette, navigated to a Web page of interest and selected text of interest. On selection of the 'SmartMark selected text' button, the portion of selected text is extracted, text words are separated from HTML tags and are sent to the respective SmartMarker. The SmartMarker compares each word in the set with a list of company names to identify company names, matches the company names to particular categories of company, and generates an annotation string and emphasis colour for each matched company name. These annotation and emphasis results are then applied to the HTML content and displayed by the Browser.

A particular example of the stages of processing of a block of text selected from a Browser's data content will now be given. Assume that a user of a Web client data processing apparatus, which includes software components implementing the present invention, selects a SmartMarker operation which identifies acronyms and generates an emphasis colour result and an annotation string for each recognised acronym. The annotation string is an expansion of the acronym. Assume that the user selects the following text content:

3 important memory types are RAM, DRAM, and ROM.

This data content in tokenized form is:

```

3
" "
important
" "
35 memory
" "
types
" "
are
40 " "
RAM,
" "
DRAM,
" "
45 and
" "
```

ROM.

The data content when filtered for a SmartMarker that requires text string tokens is:

important  
memory  
types  
are  
RAM,  
DRAM,  
and  
ROM.

The data tokens when stripped of noise are:

important  
memory  
types  
are  
RAM  
DRAM  
and  
ROM

Some tokens are then modified by SmartMarker results as follows:

```
<SPAN style='background-color:ff0000'
onmouseover='window.status="Random access memory"'
onmouseout='window.status="">RAM,</SPAN>
```

```
<SPAN style='background-color:ff0000'
onmouseover='window.status="Dynamic random access memory"'
onmouseout='window.status="">DRAM,</SPAN>
```

```
<SPAN style='background-color:ff0000'
onmouseover='window.status="Read only memory"'
onmouseout='window.status="">ROM.</SPAN>
```

When reassembled and pasted back into the Browser, the modified HTML is:

```
3 important memory types are <SPAN style='background-color:ff0000'
onmouseover='window.status="Random access memory"'
onmouseout='window.status="">RAM,</SPAN> <SPAN style='background-
color:ff0000' onmouseover='window.status="Dynamic random access
memory"' onmouseout='window.status="">DRAM,</SPAN> and <SPAN
style='background-color:ff0000' onmouseover='window.status="Read
only memory"' onmouseout='window.status="">ROM.</SPAN>
```

### SmartMarkers Design

In the preferred embodiment of the present invention, SmartMarker components are implemented as "invisible" Java Beans<sup>(TM)</sup>. Java Beans are self-contained software component classes written in the Java<sup>(TM)</sup> programming language and which can be combined with other Java Beans to perform a task when run on a computer with the support of a Java Virtual Machine. Java components are known in the art to execute as a set of objects within a Java Virtual Machine. Objects such as SmartMarker components 80,90 within the same JVM can establish communication connections with each other by calling standard Java static methods on the classes associated with other objects. Static methods are methods associated with a class rather than an instance of the class. "Invisible" in the above context means that the individual objects are represented at run time by a menu list within a palette displayed on a display\_device rather than by separate screen icons.

This palette 60 is itself implemented as a "visible" Java Bean which operates within the same JVM as the SmartMarker Java Beans 80,90. The SmartMarker operation selection palette Java Bean thus allows the user at run time to select a SmartMarker operation from a list using a mouse. The SmartMarker palette object identifies selections in response to mouse events. The use of a palette ensures that the run time representation of the SmartMarker operations takes up minimal screen space and indicates to end users that the SmartMarker operations are conceptually associated with each other and are distinct from any unrelated operations. This mechanism for item selection within a GUI and various alternatives are well known in the art.

Each SmartMarker is responsible for a specific data interpretation operation or data generation operation that processes a set of data values supplied by a SmartMarker-enabled Browser and calculates a set of results. The results may be, for example, emphasis colours and/or annotation, as will be described below. For example, an 'Average' SmartMarker determines whether numerical data values within a selected set are above, close to or below an average and then either selects emphasis colours to apply to the values to represent these categories and/or generates annotation strings interpreting individual values; whereas the 'Trend' SmartMarker calculates whether values are higher or lower than the previous value and then selects colours and/or generates an annotation result to reflect this calculation result.

The logic that defines a SmartMarker operation is coded in a SmartMarker algorithm object 90 associated with the SmartMarker Java Bean 80. A SmartMarker Java Bean represents a single SmartMarker within the palette Java Bean. Each such bean has a number of properties including title and algorithm name. This algorithm name property defines the class

name of the algorithm object 90 to be associated with the SmartMarker Java Bean. During initialisation of the SmartMarker Java Bean 80, this class name is used to construct an instance of an algorithm object 90 to be executed later when a user selects the SmartMarker Java Bean 80 and selects a set of values.

Algorithm objects 90 implement SmartMarker interface 'Algorithm' which interface defines a set of methods that will be called on an algorithm by functions within the SmartMarker transformation layer 70 after transforming the data supplied by the user interface DLL 40.

The implementation of SmartMarker algorithms 90 as separate objects from their associated SmartMarker objects 80 enables certain SmartMarkers with different properties to use the same SmartMarker algorithm object 90 for the generation of different interpretation results. Nevertheless, many SmartMarker algorithm objects 90 are only associated with a single SmartMarker object 80 (and so in alternative embodiments of the invention the SmartMarker and SmartMarker algorithm for these operations could be implemented as a single object).

To implement the Algorithm interface, an algorithm object:

- o implements method *getSmartMarkerAttributes()* to enable other parts of the SmartMarker operation model to retrieve information about the current SmartMarker (its detailed properties or attributes such as a set of direction-of-processing options displayed in the user interface which may be relevant for tabulated data values);
- o implements method *getRequiredDataTypes()* to indicate whether it requires string or numeric data, or some other data type such as links or other HTML constructs;
- o implements method *calculateResults()* to:
  - accept a value set containing a set of selected values supplied by the SmartMarker user interface DLL via the SmartMarkers transformation layer;
  - extract value objects of a specific data partition from that value set (e.g. the partition may be a row, column, or block within a table of application data values, this partition being determined by a processing Direction option within the properties of the current SmartMarker);
  - perform calculations upon these extracted values to generate a set of results;
- o optionally, an algorithm object then calls methods on some or all of the value objects to set an associated emphasis colour result



and/or calls methods on some or all of the value objects to set an associated annotation result. These *setEmphasis* and *setAnnotation* methods are implemented within a class called *ValueBase*, which is the base class for classes that represent values or sets of values passed to the algorithm (*Value*, *ValuePartition*, and *ValueSet*), and allow the algorithm to set a result on a specific value or set of values.

The above mentioned ability for the user to select different directions of processing is useful when data is organised in a structured form such as a table.

When the algorithm has completed execution of method *calculateResults*, the results are returned to the SmartMarker user interface DLL as described above.

The following methods are implemented by the SmartMarker user interface DLL (these methods are defined in mandatory application interface *SmartMarkerEnabler*, and optional application interfaces *EmphasisColor* and *Annotation*):

*SmartMarkerEnabler.startResults*

Notifies the user interface DLL that a set of results are about to be supplied, prompting the user interface DLL to start building a replacement HTML string.

*EmphasisColor.applyEmphasisColor*

Called once for each value supplied to the algorithm upon which the algorithm has called *ValueBase.setEmphasis*. Implemented by the user interface DLL to create a CSS style parameter to be added to the *<SPAN>* tag generated for the associated value in the data content in the replacement HTML.

*Annotation.applyAnnotation*

Called once for each value supplied to the algorithm upon which the algorithm has called *ValueBase.setAnnotation*. Implemented by the user interface DLL to create a DHTML string to be added to the *<SPAN>* tag generated for the associated value in the data content in the replacement HTML. In an implementation using a 'Hover Help' style, each such annotation string may subsequently be displayed in response to subsequent mouse proximity to the associated value.

*SmartMarkerEnabler.endResults*

Notifies the user interface DLL that all results have been sent, prompting the user interface DLL to complete generation of the modified HTML and to paste that HTML into the Browser's window to replace the original selected HTML.

As well as calling methods on a SmartMarker algorithm object, and implementing methods for setting annotations and emphasis colours, the respective SmartMarker object 80 has a set of properties including its name and a number of options including:

- o the type of partition of a selected data set to be analysed (e.g. single value, row, column, block),
- o a set of emphasis colours to be applied, for example red, yellow and green,
- o a set of text strings used as a key to interpret emphasis colours (i.e. each text string being applicable to one category of possible results).

Thus, a SmartMarker algorithm 90 operates upon values supplied by the Browser extension components 30,40,50 via the SmartMarker transformation layer 70 and returns a set of calculated results associated with those values which are used by the SmartMarker algorithm object 90 to generate interpretation annotation strings and emphasis colours which are returned to the Browser extension components.

Within the algorithm, each supplied value is represented by an instance of class Value. Class Value contains both the actual value and keys (such as row and column numbers) that enable the Browser extension components to recognize the source of the value. For algorithms which process numerical data values displayed in tabulated form in the Browser's window, values are grouped into partitions where each partition represents a row, column, or block. Each partition is represented as an instance of class ValuePartition. Value partitions are grouped into a single value set represented by an instance of class ValueSet.

An algorithm may return several types of results including colour changes, and annotations. To return a result for a value, the algorithm calls an appropriate method on an instance of class Value. To return a result for all values in a partition, the algorithm calls an appropriate method on an instance of class ValuePartition. To return a result for all values, the algorithm calls an appropriate method on an instance of class ValueSet.

The respective methods for generating annotation strings are:

```
Value.setAnnotation(String annotation)
ValuePartition.setAnnotation(String annotation)
ValueSet.setAnnotation(String annotation)
```

For typical operations, a SmartMarker algorithm object 90 will return the results for all values in a selected set, which the Browser extension components process and apply to the Browser content as described above. Display of annotation strings for individual data values then requires subsequent data value selection by movement of the mouse pointer to overlay the relevant data item within the Web page (which may be a cell of a table of numbers or a word within a portion of text). The correlation between mouse pointer position and individual words is handled by the Browser - for example, the Browser is instructed, by the DHTML onmouseover=' window.status="xxx"' which was added to the <SPAN>...</SPAN> group, to display a status message whenever the mouse is over the word in a tag group.

The setting of annotation strings comprises mapping each numerical result of the calculation performed by the SmartMarker algorithm to one of a set of defined result categories specific to the current SmartMarker operation. Each category has an associated text string held as a property of the SmartMarker object. For numeric SmartMarker operations, the setAnnotation method typically includes generating a text string which combines the text predefined for the identified category with the numerical result of the calculation. Examples of this generation will be described below under 'Example Operations'.

### Example Operations

As noted previously, a great many different processing operations can be implemented as SmartMarkers, including interpretation operations such as calculating for each value of a selected set whether the value is above or below the average for the set and then marking the values accordingly, or identifying and displaying a ranking of a set of data values.

US 5,896,491 (which is incorporated herein by reference) described interpretation operations in which a colour code or other emphasis is applied to individual data cells of a selected set of data cells within an application program. The colour or other applied emphasis represents the relation of each value to the other values in the set. For example, a Rank operation determines an ordering of data values by their magnitude, and the results of the operation may be presented to the user by using different shades of blue for the background of the cell containing each

data value in the selected set, with the largest value having the deepest blue background. US 5,896,491 mentions that many forms of emphasis are possible as alternatives to background colours.

To implement the Rank operation, the processing program compares all of the values in the set with each other, or compares each with a common value, to determine a ranking by magnitude, and then selects an appropriate shading or other emphasis to apply to each value to represent the relative values. US 5,896,491 disclosed implementing this by assigning values to percentile bands with all values in a given band being given the respective shading. The calculation thus preferably determines an actual ranking for each value but this is then represented to the user in a simplified, easily interpreted form.

In some circumstances, it will be highly desirable for the user to be able to confirm the actual position in the ranking of an individual value in addition to (or instead of) being presented with the very easily interpretable colour coding. For example, a user may wish for confirmation that a selected value is within the top 10% of the set of selected values, or that it is ranked third out of 50 values.

The present invention makes possible the provision of such a confirmation to the user of a Web Browser, using the mechanism for implementation of interpretation and data generation operations described in UK patent application GB 9816098.9. This mechanism is capable of presenting per-value data interpretation information to a computer user. That is, as an alternative or addition to sending to the application component an identification of the required emphasis to be applied by the Browser, a SmartMarker processing component according to the invention includes methods for providing annotation strings to the application component, the annotation strings describing the relation of a specific selected value to other values within a particular partition of a selected data set. The application component then displays an annotation string in response to user selection of a respective data value, such as selection by movement of the mouse pointer over the cell containing the value.

In association with the Rank operation described above, the annotation strings which can be returned to the application component for display alongside individual selected values may be strings which simply put into words the same information as is represented by colour codes, such as "within top 10% of values in set", "within bottom 10% of values in set", etc. Alternatively, the strings may be more detailed and give numerical information which cannot be represented by applying colour or other emphasis, such as: "ranked 3rd in row, 25 below highest".

As another example, an Average SmartMarker operation determines an arithmetic mean for a selected partition of a selected set of data and then compares individual values to the mean. The numerical results of this comparison are assigned to one of a number of categories. These categories may be, for example, the three categories "greater than mean", "within 10% of mean", and "less than mean". Each category has corresponding text stored in association therewith for inclusion within annotation text strings.

If a row of a selected data set has a mean of "M", and an individual value "N" within the selected set is determined to be greater than the computed mean for a particular partition ("Partition1", which is a row, column or block of data values - as specified within the properties of the SmartMarker by selecting the "Direction" tab of the palette) by an amount "deltaString", then the Average SmartMarker algorithm initially returns the two intermediate results M and deltaString, and in this case the intermediate results for value N are mapped to the "greater than" category. An annotation string is then generated by the setAnnotation method. This uses the intermediate results M and deltaString and stored text associated with the "greater than" result category as follows:

annotation = "N is deltaString greater than M (Partition1 mean)".

This annotation string is then returned to the SmartMarker user interface DLL for applying to the Browser's data content from which the data values were taken. An annotation text string is preferably returned for each value within the selected set of data values and the strings are stored by the Browser.

The Browser displays an annotation string in response to the mouse pointer moving over the respective data value. The Average SmartMarker has also returned to the Browser a set of annotation strings and a series of selected emphasis colours representing the different categories referred to above. The Browser has displayed on the screen the emphasis colours for the block of selected data values. The emphasis colours are obtained using the same results of the comparison between individual data values and the other values in the row, and the same categorisation. However, instead of generating annotation strings using text associated with different result categories, the emphasis operation obtains a colour associated with a result category and returns this to the application component for display.

As the mouse pointer is moved to a different data value within the same partition of the selected data set, a different one of the

annotation strings (included in the replaced HTML following generation of the strings by the Average SmartMarker) is displayed.

Thus, the methods applying emphasis colours for the Average operation provide a visually intuitive categorisation of specified partitions of a selected set of results and the methods for presenting annotation information provide supplementary information which assists user interpretation and saves the user from mental arithmetic.

An annotation operation can thus be provided in association with an emphasis-applying operation implemented by a single processing component, for processing data values supplied by a Web Browser and returning the generated results to the Browser. Alternatively, annotation may be implemented independently of emphasis.

A third example is a Compare operation, which compares each value with the first value in the row, column, or block (which partition being determined by the 'Direction' property of the SmartMarker). The positioning of the mouse pointer over a value is identified by the Browser which then presents the respective stored annotation string. The annotation strings generated by the SmartMarker object and stored by the Browser in response to selecting this Compare SmartMarker operation and selecting a set of data are of the form:

annotation = "N is deltaString greater than N1 (first in Partition1)"

where deltaString is in this case the result of subtracting the first value in the row (N1) from the current selected value (N). Partition1 is, once again, the row of the selected data set which contains the current selected value. Emphasis colours represent whether each value is greater than or less than the first value in the row.

A further example SmartMarker operation is a Trend operation, which compares each value in a selected set with the previous value in the partition (in this case, a row, ordered from left to right). Emphasis colours indicate whether each value in a row of a selected set of values is less than or greater than the previous value, and the annotation result provides alphanumeric interpretation information describing the difference between the two values. This can be very useful in aiding the interpretation of trends within data.

A further example SmartMarker operation compares each value in a selected set with a specific reference value and highlights only those results which are greater than the reference value. In this operation, data values have been selected as a block and the numerical intermediate

5 result of the calculation is a number by which each individual value exceeds the reference value. The SmartMarker assigns positive results to a "greater than" category and generates an annotation string such as is shown in the example of the Figure. For this example, no emphasis colour is applied to data values which had a negative result when compared with the reference value, simplifying the presentation to the user. Alternatively, it is possible to mark all values to remind the user which set of values was selected and to apply different emphasis colours to positive and negative results.

10 Further example operations include a Total SmartMarker which uses annotation to display the total of all values in each row or column. This is an example in which annotation can provide useful interpretation information which is relevant to the current operation and the current data values of a selected set and which is implemented independently of emphasis colours.

15 A WordCount SmartMarker uses annotation to display a count of the words in a selected block of text.

20 Statistical analysis SmartMarker and chart generator SmartMarker are examples of data generator SmartMarker operations which may also benefit from interpretative annotation.

25 It will be clear to persons skilled in the art that many different operations can be implemented using methods and mechanisms according to the present invention. In some cases the display of interpretation information will be most helpful to users if implemented together with applying per-value emphasis to represent categories of results, but in other cases the display of per-value interpretation information will be most useful if implemented as flyover text or some other media without additional emphasis. The per-value interpretation information can take many forms and even be presented via different media types as alternatives to the screen display of text strings described in detail above.

30 In view of algorithms used by SmartMarkers being implemented as software components which are executable independently of the Web Browser and are independent of individual Web pages, SmartMarker objects can be enhanced and additional SmartMarkers can be provided without having to modify the Web Browser or individual Web pages. The invention is significant in its provision of access to interpretation and analysis facilities from a Web Browser to enable semantic data processing on a Web client system of Web page content.

While the present invention has been described in detail with reference to Internet Explorer 5 and Browser extensions, it will be appreciated by persons skilled in the art that the invention is equally applicable to other Web Browsers implementing a different user interface design. The above descriptions of particular designs of user interface for a Browser extension implementation using an Explorer Bar is exemplary only, and it will be recognised that many alternative interface designs are possible within the scope of the invention.

The specific implementation described in detail presents the results of processing of Web page content as modified HTML displayed within the Browser's Web page display area, or displays the results elsewhere in the Browser's window. Alternatively, the processing components may present their results via some other mechanism (e.g. generating an audio result which is then played back via a speaker) while the original Web page is being displayed. In this case, processing results are still preferably provided to the Browser by the processing components and then the Browser or an associated presentation handler component manages presentation of different data types (text, images, audio) via appropriate presentation devices (display screen, speakers). In either case, the provision of results to the Browser does not involve downloading of a replacement Web page from a server. Such non-visual presentation of processing results is particularly relevant if the device for accessing Web page content uses non-visual interfaces for data access.



CLAIMS

1. A set of components for cooperating with a Web Browser to process the semantic data content of Web pages, wherein the Web Browser implements functions to present Web page contents to a user of a Web client data processing apparatus, and wherein the set of components are operable to run on the Web client data processing apparatus together with the Web Browser, the set of components including:

a data content transfer component, responsive to user selection of a desired semantic processing operation and user selection of a set of data items within a presented Web page, for extracting the set of data items from the Web page; and

a set of data content processing components, each for processing individual data items of a set of data items extracted from any one of multiple Web pages, for performing at the Web client data processing apparatus the selected processing operation on the individual data items of the selected set, to generate a result which is dependent on the semantics of the selected set of data items;

wherein the data content transfer component includes means for providing the generated result to the Web Browser for presentation to the user.

2. A set of components according to claim 1, wherein the data content transfer component implements functions for:

accessing the user-selected portion of the markup language content of a Web page, separating individual data items from markup tags within the selected portion, and passing the separated data items to one of said set of data content processing components; and,

in response to receipt of a generated result from said data content processing component, generating a modified version of the portion of the markup language Web page content, and providing said modified portion to the Web Browser to replace the extracted portion of the presented Web page.

3. A set of components according to claim 2, wherein the separating of data items comprises parsing the selected portion of the markup language content to separate data items from markup tags, and parsing the separated data items to remove any extraneous characters.

4. A set of components according to claim 2 or claim 3, wherein the generating of a modified version of the portion of the markup language Web page content comprises constructing a markup language separator tag pair for each data item for which results are returned, separately wrapping each of said data items using said tag pair and applying the results to the wrapped data items.

5. A set of components according to claim 4, wherein data content processing components of said set generate emphasis results including one or more style attributes, and wherein applying said emphasis results to a respective wrapped data item includes adding to the respective tag pair a style parameter including the respective one or more style attributes.

6. A set of components according to claim 5, wherein said style attributes include a background colour predefined to correspond to a particular emphasis result.

7. A set of components according to claim 4, wherein data content processing components of said set generate annotation results including annotation strings, and wherein applying said annotation results to a respective wrapped data item includes adding a segment of dynamic HTML or other scripting language to the respective tag pair which segment activates presentation of a respective annotation string in response to mouse pointer positioning.

8. A set of components according to claim 1 or claim 2, for use with a Web Browser supporting a Browser extension architecture for linking to executable program code which is separate from the Browser, wherein the data content transfer component is implemented as one or more Browser extension components.

9. A set of components according to claim 2, wherein the markup language is HTML.

10. A method for processing the semantic data content of Web pages presented via a Web Browser to a user of a Web client data processing apparatus, the method including the following steps performed on the Web client data processing apparatus:

in response to user selection of a desired semantic processing operation, and user selection of a set of data items within a presented Web page, extracting the set of data items from the Web page; and

performing the selected processing operation on individual data items of the selected set to generate a result which is dependent on the semantics of the selected set of data items, using a data content

processing component which is capable of processing data items of a set of data items extracted from any one of multiple Web pages;

5           providing the generated result to the Web Browser for presentation to the user.

11.   A computer program for running on a Web client data processing apparatus and cooperating with a Web Browser to process the semantic data content of Web pages, wherein the Web Browser implements functions to  
10   present Web page contents to a user of the Web client data processing apparatus and wherein said computer program comprises

        a data content transfer component, responsive to user selection of a desired semantic processing operation and user selection of a set of data items within a presented Web page, for:

15           extracting the set of data items from the Web page;

        providing said set of data items to a respective one of a set of data content processing components running on said Web client which data  
20   content processing component is capable of performing the selected processing operation on individual data items extracted from any one of multiple Web pages to generate a result which is dependent on the semantics of the individual data items of the selected set; and

25           providing the generated result to the Web Browser for presentation to the user.

12.   A Web client data processing system including a set of components according to claim 1.

30   13.   A software component for cooperating with a Web Browser, the software component implementing functions to control the operation of a data processing apparatus on which it runs to perform the following steps in response to user selection of a desired data processing operation and  
35   user selection of a portion of a Web page presented via the Browser to a user of the data processing apparatus:

        extracting from the Web page the selected portion of the markup language Web page content;

40           separating data items from markup tags within the selected portion;

        passing the separated data items to a data processing component to perform a processing operation on the individual data items; and

in response to receipt by said software component of a result of said processing operation, generating a modified version of the portion of the markup language Web page content; and

providing said modified portion to the Browser to replace the extracted portion of the presented Web page.

14. A software component according to claim 13, wherein the separating of data items from markup tags comprises parsing the selected portion of the markup language content of the Web page to separate data items from markup tags, and parsing the separated data items to remove any extraneous characters.

15. A software component according to claim 13 or claim 14, wherein the generating of a modified version of the portion of the markup language Web page content comprises constructing a markup language separator tag pair for each data item for which results are returned, separately wrapping each of said data items using said tag pair and applying the results to the wrapped data items.

16. A software component according to claim 15, wherein data content processing components of said set generate emphasis results including one or more style attributes, and wherein applying said emphasis results to a respective wrapped data item includes adding to the respective tag pair a style parameter including the respective one or more style attributes.

17. A software component according to claim 16, wherein said style attributes include a background colour predefined to correspond to a particular emphasis result.

18. A software component according to claim 15, wherein data content processing components of said set generate annotation results including annotation strings, and wherein applying said annotation results to a respective wrapped data item includes adding a segment of dynamic markup language to the respective tag pair which segment activates presentation of a respective annotation string in response to mouse pointer positioning.

19. A method for assisting a user of a Web client data processing apparatus with interpretation or analysis of the data content of Web pages presented to the user via a Web Browser, the method including the following steps performed on the Web client data processing apparatus:

in response to user selection of a desired data processing operation and user selection of a portion of a Web page presented via the

Browser, extracting from the Web page the selected portion of the markup language Web page content;

5            separating data items from markup tags within the selected portion;

          passing the separated data items to a data processing component to perform a processing operation on the individual data items; and

10           in response to receipt by said software component of a result of said processing operation, generating a modified version of the portion of the markup language Web page content; and

          providing said modified portion to the Browser to replace the extracted portion of the presented Web page.

15           20. A computer program product comprising computer readable program code recorded on a computer readable recording medium, the program code including a software component according to claim 13.

20           21. A Web client data processing system including a software component according to claim 13.

          22. A data processing component for interoperating with a software component according to claim 13, including:

25                program code for receiving the separated data items and formatting and/or filtering the data items for processing;

30                program code for performing a processing operation on the formatted and/or filtered individual data items to generate a result; and

          program code for returning the result to said software component.

35           23. A data processing component according to Claim 22, wherein said generated result includes an annotation string and/or a style attribute for inclusion in a modified version of the portion of markup language Web page content.



INVESTOR IN PEOPLE

Application No: GB 9923173.0  
Claims searched: 1-12

Examiner: Mike Davis  
Date of search: 27 April 2000

## Patents Act 1977 Search Report under Section 17

### Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.R): G4A (AUIDB, APX)

Int Cl (Ed.7): G06F

Other: Online: WPI, EPODOC, JAPIO

### Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	GB 2329492 A (MICROSOFT) eg abstract	-
X	WO 98/18087 A1 (BT) eg abstract, page 19 line 10 to page 25 line 15, and page 32 lines 1-3	1,10,11 at least

X Document indicating lack of novelty or inventive step  
Y Document indicating lack of inventive step if combined with one or more other documents of same category.

& Member of the same patent family

A Document indicating technological background and/or state of the art.  
P Document published on or after the declared priority date but before the filing date of this invention.

E Patent document published on or after, but with priority date earlier than, the filing date of this application.



INVESTOR IN PEOPLE

Application No: GB 9923173.0  
Claims searched: 13-21

Examiner: Mike Davis  
Date of search: 14 August 2000

**Patents Act 1977**  
**Further Search Report under Section 17**

**Databases searched:**

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.R): G4A (AUSB, APX)

Int Cl (Ed.7): G06F

Other: Online: WPI, EPODOC, JAPIO

**Documents considered to be relevant:**

Category	Identity of document and relevant passage	Relevant to claims
X,E	GB 2346238 A (IBM) whole document	13,19-21 at least
X	WO 98/18087 A1 (BT) eg abstract, page 19 line 10 to page 25 line 15, and page 32 lines 1-3	"

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.